AMENDMENTS TO THE CLAIMS

1. (Currently Amended) A data structure that is stored on a computer-readable medium comprising:

a sorted portion that contains a plurality of entries that are sorted into an order;

an unsorted portion that contains a plurality of entries that have not been sorted; and

a boundary that separates the sorted portion and the unsorted portion;

wherein the sorted portion of the data structure ~~may be searched~~ is searchable with O(logN) performance while an entry is added to the unsorted portion.

2. (Currently Amended) The data structure of claim 1, wherein the sorted portion ~~may be searched~~ is searchable with a binary search.

3. (Currently Amended) The data structure of claim 1, wherein the unsorted portion ~~may be searched~~ is searchable with an incremental search.

4. (Original) The data structure of claim 1, wherein the data structure may be sorted to form a new sorted portion that comprises the plurality of entries of the sorted portion and the plurality of entries of the unsorted portion, and the plurality of entries of the new sorted portion are sorted into an order.

5. (Original) The data structure of claim 1, wherein the data structure is associated with an occurrence model used in designing a circuit.

6. (Original) A method of using a container that comprises a sorted portion that contains a plurality of entries that are sorted into an order, an unsorted portion that contains a plurality of entries that have not been sorted, and a boundary that separates the sorted portion and the unsorted portion, the method comprising:

receiving a search request that comprises a requested value;

searching the sorted portion of the container for the requested value with O(logN) performance;

adding an entry to the unsorted portion during the searching; and

returning a stored value of the container if there is a match of the stored value and the requested value.

7. (Original) The method of claim 6, wherein when there is not a match, the method further comprises:

returning a null value that indicates that there is no match with the requested value.

8. (Original) The method of claim 6, wherein when there is not a match, the method further comprises:

adding an entry to the unsorted portion corresponding to the search request.

9. (Original) The method of claim 6, wherein when there is not a match, the method further comprises:

determining whether unsorted items in the container exceed a predetermined threshold;

performing a sort operation on the container, if the predetermined threshold is exceeded, thereby forming a new sorted portion that comprises the plurality of entries of the sorted portion and the plurality of entries of the unsorted portion, and the plurality of entries of the new sorted portion are sorted into an order.

10. (Original) The method of claim 9, further comprises:

searching the new sorted portion of the container for the requested value; and

returning a stored value of the container if there is a match of the stored value and the requested value.

11. (Original) The method of claim 10, wherein searching the new sorted portion comprises:

searching with O(logN) performance.

12. (Original) The method of claim 6, wherein when there is not a match, the method further comprises:

searching the unsorted portion of the container for the requested value; and

returning a stored value of the container if there is a match of the stored value and the requested value.

13. (Original) The method of claim 12, wherein the unsorted portion may be searched with an incremental search.

14. (Original) The method of claim 6, wherein when there is not a match, the method further comprises:

determining whether a size of the unsorted portion is zero;

adding an entry to the unsorted portion corresponding to the search request if the unsorted portion is not zero.

15. (Original) The method of claim 14, wherein the size of the unsorted portion is zero, the method further comprises:

determining whether the requested value of the search request is greater than the value of the last entry of the sorted portion;

adding an entry to the unsorted portion corresponding to the search request if the requested value of the search request is not greater than the value of the last entry of the sorted portion;

adding an entry to the sorted portion corresponding to the search request if the requested value of the search request is greater than the value of the last entry of the sorted portion.

16. (Original) The method of claim 6, further comprises:

using the container in an occurrence model in designing a circuit.

17. (Original) A computer program product having a computer-readable medium having computer program logic recorded thereon for inserting a new value into a container that comprises a sorted portion that contains a plurality of entries that are sorted into an order, an unsorted portion that contains a plurality of entries that have not been sorted, and a boundary that separates the sorted portion and the unsorted portion, the computer program product comprising:

code for searching the sorted portion of the container for the new value with O(logN) performance;

code for searching the unsorted portion of the container if no match is found in the search of the sorted portion with O(N) performance; and

code for inserting the new value into the container if no match is found in the search of the unsorted portion.

18. (Original) The computer program product of claim 17, wherein the code for inserting comprises:

code for determining whether to insert the new value in the sorted portion or the unsorted portion of the container.

19. (Original) The computer program product of claim 17, further comprises:

code for sorting the unsorted portion and merging the sorted portion and the sorted unsorted portion into a new sorted portion, wherein the code for sorting is operative when the unsorted portion exceeds a predetermined criteria; and

code for searching the new sorted portion of the container for the new value with O(logN) performance.

20. (Original) The computer program product of claim 17, further comprises:

code for a circuit design.

21. (Currently Amended) A computer system for managing data objects, comprising:

memory means for storing said data objects;

means for identifying a boundary within said memory means for storing, wherein data objects stored in a first portion of said memory ~~storing~~ means defined by said boundary are stored in an ordered manner and data objects stored in a second portion of said memory ~~storing~~ means defined by said boundary are stored in an unordered manner; and

means for searching said first portion according to O(logN) performance to locate an identified object.

22. (Original) The computer system of claim 21 further comprising:

means for searching said second portion for said identified object according to O(N) performance.

23. (Original) The computer system of claim 21 further comprising:

means for adding said identified object to said second portion when said means for searching said first portion and said means for searching said second portion do not locate said identified object.

24. (Original) The computer system of claim 21 further comprising:

means for merging data objects in said second portion into said first portion in an ordered manner; and

means for resetting said boundary in response to said means for merging.

25. (Original) The computer system of claim 24 wherein said means for merging is operable when a number of data objects in said second portion reaches a predetermined amount.